LEARN TO PROGRAM GTIA GRAPHICS

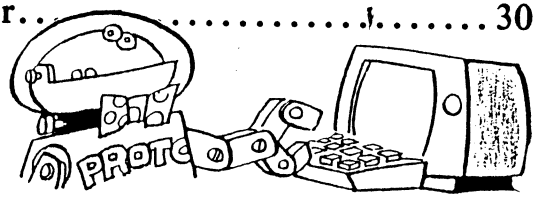
TRICKY TUTORIAL™ no. 9

```
87,34:POKE 207,99:NEXT I
X+27,Y+88:SOUND 0,2,8,10
,12:FOR I = LEN(ABS(INT(X
N GOSUB 50:READ BOOK:BOOK
Z = PEEK(242):LIST 800,12
```

EDUCATIONAL SOFTWARE IN

TABLE OF CONTENTS:

The GTIA Software.....3
Introduction.....4
Gtiatest.....7
Demo.....9
Mode 9.....12
Picture.....13
Nine.....14
Mode 10.....15
Sticks.....15
Tredmill.....18
Cylinder.....18
Blakhole.....18
Mode 11.....19
Eleven.....20
The Price You Pay for Art.....22
Window.....23
Player-Missiles.....24
Software Compatability.....26
The GTIA Editor.....30



The GTIA Program Listings

Gtiatest.....33
Demo.....34
Picture.....36
Nine.....36
Sticks.....37
Tredmill.....38
Cylinder.....38
Blakhole.....39
Eleven.....40
Window.....40
Gedit.....40

TRICKY TUTORIAL #9
GTIA GRAPHICS

by Sheldon Leemon
with Stan Gilbert

HOW TO LOAD

Tape....

With the BASIC cartridge in place, turn the computer on. Place the tape in your recorder, label side up. Make sure the tape is rewound and the counter is reset to zero. Push PLAY on the recorder, type CLOAD and press RETURN twice. If the program won't start to load, try positioning the tape forward or backwards a little. When the READY prompt appears, type RUN and press RETURN. We recommend you write down the number of your recorder's counter on the next page as each program example starts as this will make it easier to find each part later on.

Disk....

To LOAD and RUN the disk, first turn on your disk drive. When the busy light goes out, place the disk in the drive. With the BASIC cartridge in place, turn on the computer and the program will LOAD and RUN each part by itself.

ATARI is a registered trademark of Warner Communications, Inc.
Manual & Software © 1983 by Educational Software,Inc.

Any defective tapes or disks should be returned to:

Educational Software Inc.
4565 Cherryvale Ave.
Soquel, CA 95073
(408) 476-4901

The GTIA Software

The following programs are included with this tutorial. At the top of each page in the manual is the name of the program which you should be running. Tape users should note the value of the counter as each program begins to load on the lines below. This will make it easier to find each program when you later run these examples.

PROGRAM	STARTS AT:
1 GTIA.LGO	0
2 GTIATEST	
3 DEMO	
4 PICTURE	
- OUR WORKSHOP	
- OUR ACCOUNTANT	
- OUR SECRETARY	
- OUR BOSS	
5 NINE	
6 STICKS	
7 TREDMILL	
8 CYLINDER	
9 BLAKHOLE	
10 ELEVEN	
11 WINDOW	
12 GEDIT	

INTRODUCTION

As the greatest living authority on ATARI Home Computers, I get lots of letters (some from as far away as Deep Space) asking all kinds of questions about the mysteries of the computer. Lately, I've been getting so many of them about the new GTIA chip that not even my pal Prototype can keep up with them. That's why I have come up with another of my Tricky Tutorials to reveal all of its secrets (or at least enough of them for you to get started). So please, no more letters, especially from the joker whose Intergalactic Spacegrams have been arriving Postage Due. Here are the questions, and the answers.

WHAT THE HECK'S A GTIA ANYWAY?

I could tell you that it was a souped-up (GT) model of the CTIA, but that would raise the inevitable question, What the heck's a CTIA? To answer that question we've got to go to the museum of ancient microcomputers. Before ATARI, most microcomputers used a simple bit-mapped display—that means the screen display was divided into a grid of points of light (pixels), and a whole bunch of binary digits (bits) were kept in memory to tell the main processor whether to turn the point of light on (represented in memory by the number 1), or off (represented by the number 0). This kind of display scheme can be easily handled by a single central microprocessor while it is doing all of the rest of the computing at the same time. The ATARI computer not only has many different graphics modes, regular and split screen, up to 256 different color-luminance combinations, and player-missile graphics, but it also has four-voice sound through the TV speaker as well as an internal speaker, joystick, paddle and light pen controllers....you get the idea. That would be a lot of work for the 6502 micro- processor to do, in addition to its regular calculating chores. Even if it could handle it all, letting the main processor do all of the work would make programming the ATARI REALLY complicated.

Fortunately for us, ATARI came up with a neat solution by creating three silicon helpers for the main micro- processor. These little guys are called, ANTIC, POKEY, and CTIA/GTIA. ANTIC controls access to memory so that each chip gets his turn, counts the vertical lines of the screen display and checks for light pen inputs, keeps track of fine scrolling, and generally controls the display timing. Despite his name, POKEY is not slow. He watches for input from the parallel ports (where you plug in the joysticks and paddles) over 60 times per second, as well as oversees the system timers and sound, and generates random numbers. So far, the Tricky Tutorial series has covered the hi-jinks of ANTIC in the Display List and Scrolling Tutorials, and POKEY has gotten his due in the Sound and Music package. Here at last, that schizophrenic bit-brain CTIA/GTIA will get some of the recognition he so richly deserves (we're so fond of him we can't bear to refer to CTIA/GTIA as "it").

WHAT'S HE DO?

Actually, some of his functions have been dealt with elsewhere in the series, because he controls Player-Missile objects and their related priority, size, collision detection, and horizontal position registers. But the functions we are going to talk about here have to do with Color-Luminance control of ordinary playfield graphics--the kind that you control from BASIC with the garden variety COLOR, PLOT and DRAWTO statements while in drawing modes 3 through 11. It is this little devil who allows you to display those wonderful color pictures on your home TV set, without the need for a separate RF modulator like most other computers. One of the nice side-effects of this is that you can directly video-tape the computer output with a home VCR. That means that you can save your really high game scores for posterity, or use your ATARI for titling your home video-tapes. I even know one TV studio technician who has used it to create "crawls"--the moving line of text broadcast at the bottom of the screen for special announcements.

GTIA.LGO

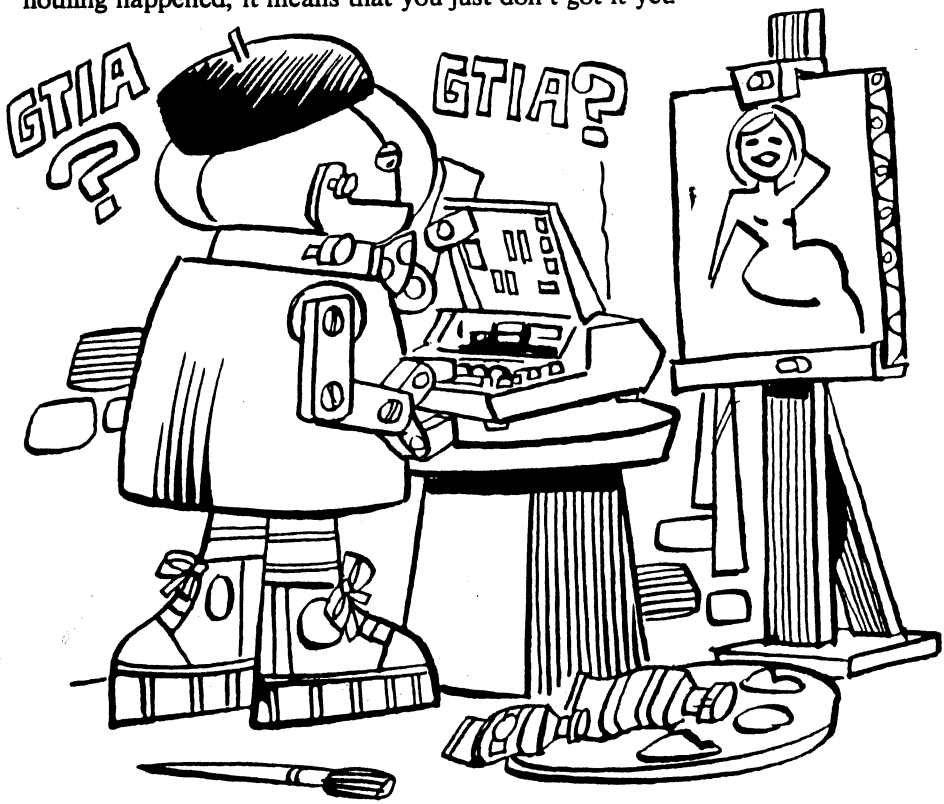
WHAT DO YOU MEAN, MODES 3 THROUGH 11?

Everyone who has read the good ole' ATARI BASIC Reference Manual (raise your hands) knows that ATARI has Graphics Modes 0-8. So where did 9, 10, and 11 come from? Well, that's where the story of CTIA/GTIA's split personality comes in. The original design for the ATARI computers included Graphics modes 0-11, under the loving guidance of the GTIA chip, or Graphics Television Interface Adapter. But for some reason, they couldn't get the Adapter to adapt right for graphics modes 9, 10, and 11 on American-style TV sets.

So, in order to get some version of the ATARI computer out on time (8 graphics modes were enough to start with, weren't they?), a somewhat simpler version, the Color Television Interface Adapter, without modes 9-11 supported, was shipped as standard equipment on American ATARI computers made before November, 1981. Oddly enough, the original European version of the GTIA, made for computers sold in foreign countries (such as Germany and England) using the European PAL television standard, has worked just fine all along. But ATARI forgot to tell them about it, so the people who bought ATARIs in those countries have had the GTIA for a couple of years, without knowing about it. Hmm, come to think of it, there still is no updated BASIC Reference Manual mentioning those modes to American buyers either. Lucky for you the Professor reads his mail!

DO I GOT IT?

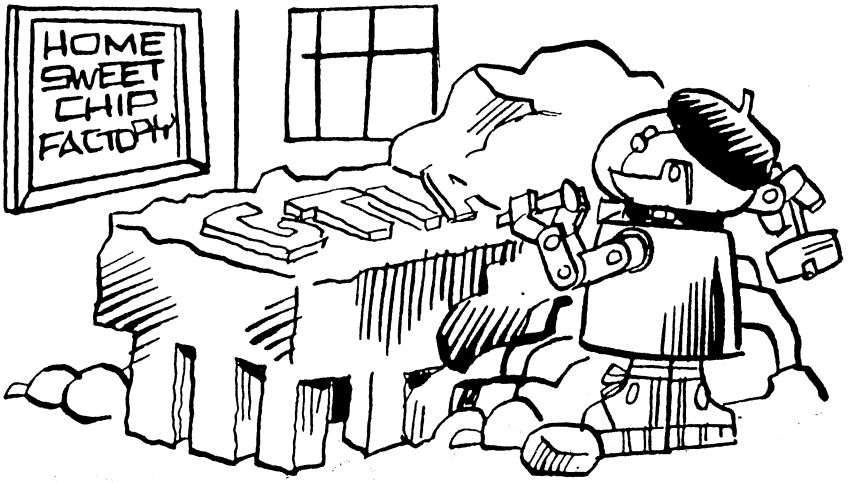
To tell if they have a GTIA chip, some people rely on signs and omens, like a little 'G' sticker on the bottom of their computer case. Don't waste your time with such superstitious nonsense. The first program following the logo program tests for the presence of a GTIA chip. This program is called GTIATEST and will be self explanatory. If you have the GTIA chip, the program will automatically chain to a program called RAINBOW which will give you a taste of the light and color possibilities available to the ATARI computer equipped with a GTIA chip. The truth is, you can tell if you have a GTIA chip with one little POKE. Put your BASIC cartridge in, fire up the computer, and enter the command POKE 623,64. Did your screen turn black and the writing become all funny and illegible? Congratulations, you got it! If nothing happened, it means that you just don't got it yet.



I DON'T GOT IT! HOW DO I GET IT?

There are two ways to do this, the easy way and the hard way. The easy way, as always, costs more. You can let your authorized ATARI dealer install the chip for you. This will usually cost $25 for the chip, and almost as much for the labor. For you diehard cheapskates (or incurable hackers), it is possible to buy the chip and install it yourself. Usually you can get instructions on installation when you buy the chip.

Remember, installing the chip yourself will void your warranty. Faulty installation could cause your computer to go Kablooey! But if you have already taken the little fellow apart, and you feel confident that you can plug a chip into a socket, you might want to take the risk. The intermediate solution to all of this is to find an experienced computer hobbyist who has already installed some – maybe by joining your local ATARI computer club (ATARI User Group Support has a list of over 200 of them, so there must be one near you). Then 'finagle' him or her into doing it for you. Use your judgement, and pick the solution best for you.



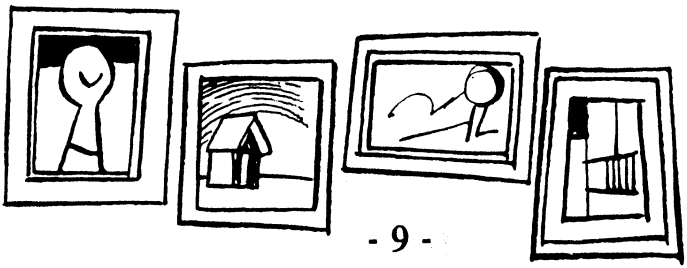
I GOT IT! I GOT IT!
NOW WHAT DO I DO WITH IT?

Remember Graphics modes 9, 10, and 11, that I told you about earlier? Well now you're going to find out what they do.

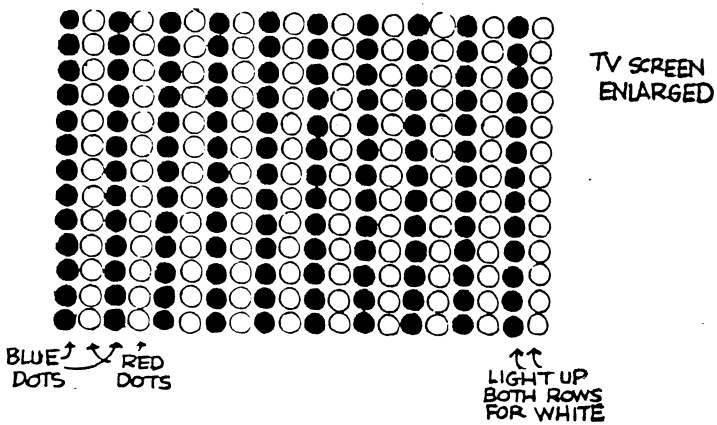
DEMO

While the next program, DEMO, is running, Proto will explain some of the finer points of resolution, indirection, and artifacting. Again, you're greeted with a menu of choices. Pick what you'd like to see (by pressing the number which corresponds to that choice), and Proto will show you the basic concepts for that item. Proto will explain how the resolution varies from mode 5 and 7 to the GTIA modes. Proto's lecture on color indirection will put stars in your eyes. He'll also show you how to change colors the old-fashioned way by redrawing them, just so you'll be properly impressed. His last act will be to demonstrate artifacting on the ATARI. So if you want to know what is odd and what is even then press choice number three.

Artifacting is a peculiarity of displays generated by the ATARI on color CRTs in the GRAPHICS 0 mode. Each character on the screen is 'painted' in an 8x8 dot matrix. Each dot is called a PIXEL. If two pixels adjacent to one another are illuminated then you witness a white image on the screen. If, however, every other pixel is turned on then you will see a red or blue image, depending on whether or not the image starts with the first pixel (ODD) or second pixel (EVEN). Stated in simpler terms, when odd pixels are illuminated without adjacent horizontal pixels being turned on the image will appear blue. The image will be red if even numbered pixels are turned on with adjacent horizontal pixels turned off. When Proto has finished each topic, he'll return you to the menu. To exit, press '4' to quit. When you're ready, RUN the next program, PICTURE.



DEMO



The function of modes 9, 10, and 11 is to allow you to put more of the 256 possible color-luminance combinations on the screen at the same time. Graphics mode 9 allows you to use up to 16 shades of the same color at one time. Graphics mode 11 allows you to display up to 16 colors at the same luminance value at one time. And Graphics mode 10 allows you up to 9 distinct combinations of any color with any luminance at the same time. Since I know that you are anxious to see how this works, I've provided the following short program for you to key in when you're ready.

```
10 GRAPHICS 9
20 SETCOLOR 4,12,0
30 FOR I = 0 TO 15: COLOR I
40 PLOT 4*I+5,2:DRAWTO 4*I+5,190
50 DRAWTO 4*I+6, 190:DRAWTO 4*I+6,2
60 NEXT I
70 GOTO 70
```

You should see 16 shades of green bars on the screen. If you want to see 16 colors at once, make the following changes:

```
10 GRAPHICS 11
20 SETCOLOR 4,0,4
```

Pretty exciting, eh? Take a rest, and when you come back I'll have you RUN some examples to see what these modes are all about.

MODE BY MODE WITH THE GTIA

While I have tried to give a general overview of the GTIA modes, each one has its own talents and special characteristics which we will now examine.

MODE 9
--A shade of difference

Mode 9 allows you to have 16 different shades of the same color on the screen at one time. You should notice that this is a big difference from the normal graphics modes. In those modes, the LUMINANCE value of a color is an even number from 0 to 14. That means that there are only 8 levels of brightness available. With mode 9, you have twice as many levels. This means, for one thing, that there are now 256 shades to play with (16 colors with 16 levels of brightness for each color). By drawing stripes on the screen each with a different shade of one color, and then using Display List Interrupts to change the Hue value every few lines, we can actually display all 256 colors on the screen at once.

We will summarize the material given above so as to provide practical operating instructions for using Mode 9. Step by step, these are:

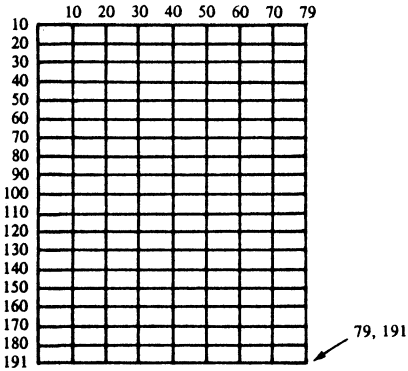
1) Use a GRAPHICS 9 command to set the mode.

2) Use a SETCOLOR statement to select the color to be used. The register used to specify this color is the one which holds the background color read at location 712. The SETCOLOR statement should be in the form SETCOLOR 4,C,0 where C is the color value from 0 to 15 you wish to use.

3) When you wish to PLOT or DRAWTO, use the COLOR command to specify the shade of the color you wish to use. COLOR 0 selects the darkest shade and COLOR 15 the lightest. You will notice that in this context only, the COLOR command corresponds to the actual shade, and not a color register number which contains the value of the color information.

4) After selecting a color, PLOT and DRAWTO normally. The vertical resolution is the same as Mode 8, but the horizontal resolution is only 1/4 as fine. Therefore, the limits of legal values for the horizontal coordinate will be 0-79, while those for the vertical component will be 0-191 (remember, there is no text window). Anything between 0,0 and 79,191 is kosher.

PICTURE



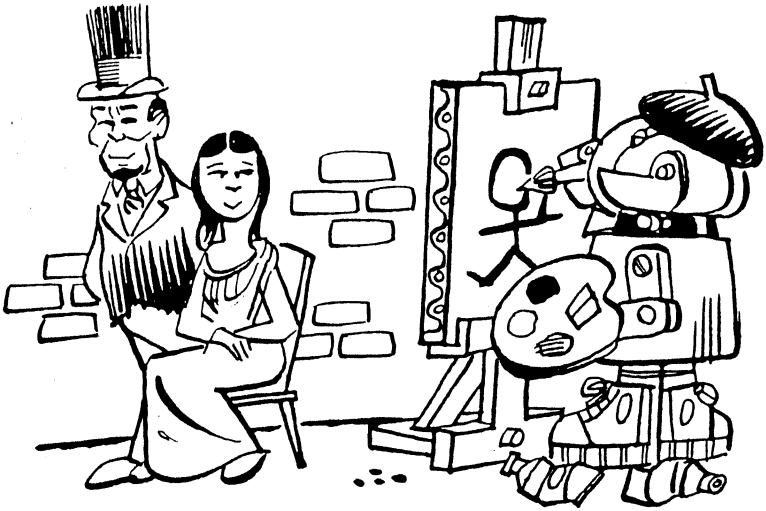
Now that you know how to use mode 9, you may be wondering what it is good for. Well, because of the expanded gray-scale available, this mode is good for representing shapes that use shading to achieve a three-dimensional quality. A simple program to represent a cylinder shape is presented below:

```
100 GRAPHICS 9
120 SETCOLOR 4,15,0:REM Color is gold
130 FOR I=25 TO 49
140 C=I+(74-I-I)*(I>37)-22
150 D=INT(SQR(144-(I-37)*(I-37)))/2
160 COLOR 15-C:PLOT I,32-D:DRAWTO I,32+D
170 COLOR C:DRAWTO I,155+D
180 NEXT I
190 GOTO 90:REM maintain GRAPHICS mode
```

A more advanced application is the display of digitized images such as those in the PICTURE program. PICTURE is a program containing some useful techniques to display rather complex screens. Cassette users may only view these pictures in order. To load each picture, press RETURN when the computer's internal speaker beeps. If a picture fails to load, the tape must be repositioned for another attempt. Disk users make their selections from a menu of choices. The first screen is a drawing done using the GTIA Editor provided in this tutorial. It shows Professor Von Chip's little software factory at the edge of the enchanted redwood forest near the sea. It is presented in GRAPHICS 10 which utilizes 8 colors. The PICTURE program does not interpret the screen data to determine what GRAPHICS mode is being used. The programmer must know in advance, otherwise, the same data may be presented in a different GRAPHICS mode and the display may actually be unrecognizable. After you are all done admiring the pictures, press any key to return to the PICTURE menu.

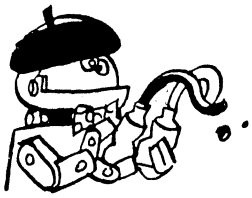
NINE

The remaining four screens are digitized from photographs and are displayed using GRAPHICS 9. That means that the video images of the photographs were analyzed by the computer and broken down into individual "PIXELS" that, when viewed collectively, appear as a recognizable image. This program can be modified to display any picture you draw using the GTIA Editor. Complex image presentations may be practical for sales or instructional efforts that utilize the computer. Of course, we are limited by the size of the pixels, which only give us a resolution of 80 dots across. Still, some very recognizable images can be displayed. Examples of these are LINCOLN and MONA LISA. These images were first broken down to gray-scale values for each dot in a 64X64 matrix. These values were averaged to reduce the original 32 levels of brightness to the 16 levels available in this mode.



Next run the program, NINE to see an image created using the several shades available in GRAPHICS 9 to create a 'modeled' image. The code for this program and most others can be found at the back of this manual. This program uses information stored in DATA statements to create the image of a pipe. The pipe is drawn using regular PLOT and DRAWTO statements. Our little friend Prototype, situated on the pipe, is an example of player-missile graphics. As you can see, there is no conflict between using GRAPHICS 9 and player-missiles simultaneously. Press the SPACE BAR to run the next program, STICKS.

STICKS



MODE 10
--The Jack of All Trades

Mode 10 has the potential for being one of the most important of all of the ATARI graphics modes. The reason for this is that it combines the best of both worlds–it gives you an extended selection of the number of colors you can get on the screen without using Player-Missile graphics. As a matter of fact, it's hard to come up with a situation when you will need more than the 9 colors it offers. Yet unlike the other GTIA modes, it allows you the benefits of color indirection. This means that you can select each of these 9 colors from one of the 128 color/hue combinations available, and that you can change the color of any object on the screen instantly, with just one SETCOLOR or POKE command!

STICKS is a simple example of how to manipulate these color registers to give the impression of motion in the image. Vertical lines are drawn on a GRAPHICS 10 screen. Using different shades of the same color in the color registers and then shifting each color register to the next gives the impression of motion. The following code shifts the color registers:

4260 A=PEEK(712)
this just stores the last color register in 'A'

4270 FOR I=721 TO 708 STEP -1
this sets up the loop to move each color register down

4280 B=PEEK(I-1):POKE I,B
looks at the previous register, sticks it into the current register

4290 NEXT I:POKE 707,A
finishes shifting colors and moves 'saved' last color into first color register.

STICKS

Because of the differences in possible color/hue combinations, the operating instructions for Mode 10 are slightly different than for its cousins, Modes 9 and 11. To use this mode, you must:

1) Give the GRAPHICS 10 command.

2) Select a color register to draw with using the COLOR command. Here, the COLOR command refers to the register which contains the color value, and not the color value itself. Therefore, the range of values used with the command is COLOR 0 to Color 8. While other numbers will be accepted by BASIC, these numbers will just be interpreted as one of the 9 values from 0 to 8.

Of course, before you use a color register, it would help to know what actual color value it contains, and how to change that value. In the case of the 5 normal playfield registers, the kindly Operating System fills in default values–the familiar orange, blue, pink, light green, and black. These can be controlled by the SETCOLOR command. But Mode 10 uses the Player-Missile color registers to give you four extra selections. These registers have no SETCOLOR command. Moreover, the Operating System does not even have the decency to set up a default color. Like the song says, use these registers before changing the color, and you're going to "Paint it Black". Fortunately, you can change these registers with a simple POKE.

STICKS

Register 0 corresponds with hardware location 704 decimal. Therefore, COLOR 0 will set the color to be PLOTted according to what's in 704. To calculate this number use the following formula:

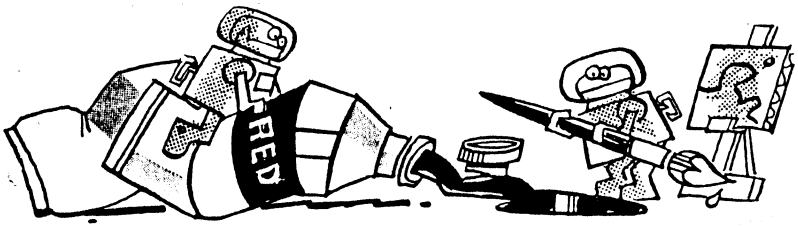
N = (16 * color value) + luminance value

For example, to set color register 0 to the background blue of a Mode 0 screen you would want the equivalent of the value generated by the statement SETCOLOR x,9,4. Since the color value is 9, and the luminance value is 4, the number to POKE into 704 to get this same color is (16*9)+4, or 148. Likewise, an orange (color 2 on the previous chart) with a luminance value of 8 would be produced by POKEing in the number (2*16)+8, or 40.

This type of color POKEing works just the same on the normal playfield colors. These registers are located directly after the Player-Missile registers in memory. The PM color registers are 704-707, and the playfield registers are 708-712. To orient you a little better, 712 holds the background black of the map modes, and the black frame of Mode 0. If you POKE 712,14, the black frame will turn white, just like with a SETCOLOR 4,0,14. Location 710 usually holds the blue background of Mode 0. If you POKE 710, 0, it will turn black.

To summarize, the COLOR statement sets up a color register for you to PLOT in. The color of these registers can be changed as follows: Registers 0-3 by POKEs, Register 4-8 by POKEs or by use of the SETCOLOR 0-4 command.

3) Whew! Now that you've set up the color register and determined its color, you're ready to PLOT and DRAWTO. The screen limits are the same as Modes 9 and 11, as discussed previously.



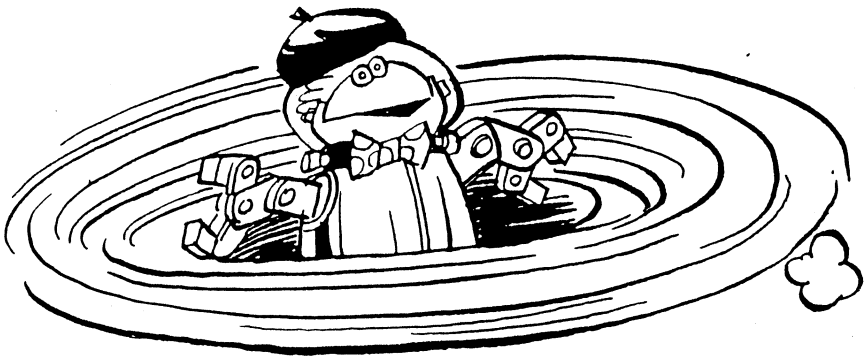
TREDMILL/CYLINDER/BLAKHOLE

When you view the next program you will see an example of why we might want to do something like this. The very same appearance of motion from STICKS is applied to TREDMILL and makes it look like PROTO is forever caught on a tredmill. If he could only reach that 'BREAK' key! (You should use the 'Q' key to exit this program).

CYLINDER is another variation on the same theme. The major difference is that we have set most of the color registers to the same color. We then proceeded to draw the cylinder using each color register. However, one color register is a lighter shade, almost white. Now when we shift the color registers the light color moves to the next register and the appropriate set of vertical lines change color. This makes the cylinder appear to rotate. Notice that the vertical lines that form the inside top of the cylinder only extend to the outside top edge. This creates the appearance of three dimensional motion. Press 'Q' to exit.

Again, a variation of the same theme is evident in BLAKHOLE. The spiral is drawn using different color registers. Shifting the colors animates the spiral and PROTO appears to be caught in a hopeless trap. The use of PLAYER MISSILE GRAPHICS is permitted in GTIA GRAPHICS modes, however, there are some restrictions! We'll get into those shortly.

GRAPHICS 10 is really very versatile. It can be used to make the hypnotic, moving displays you saw in the program STICKS. On the other hand, it can also be used much like Mode 9 to create beautiful static displays such as the one of my little factory in the program PICTURE.



MODE 11
--A Pixel of a Different Color

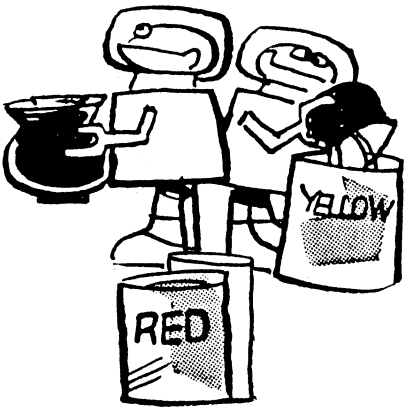
Mode 11 is completely analogous to Mode 9, except it offers 16 colors at one level of brightness. The instructions for using it are almost identical to those for Mode 9:

1) Use a GRAPHICS 11 command to set the mode.

2) Use a SETCOLOR statement to select the color to be used. That statement should be in the form SETCOLOR 4,0,B where B is the brightness level which you wish to use. Note that in this mode, you are restricted once again to 8 brightness values, so make sure that B is an even number in the range from 0 to 14.

3) When you wish to PLOT or DRAWTO, use the COLOR command to specify the color which you wish to use. These colors correspond exactly to the shades which you normally select using the second value of the SETCOLOR command. To refresh your memory, these values are:

Gray	0
Gold	1
Orange	2
Red-Orange	3
Pink	4
Purple	5
Purple-Blue	6
Azure Blue	7
Sky Blue	8
Light Blue	9
Turquoise	10
Green-Blue	11
Green	12
Yellow-Green	13
Orange-Green	14
Light orange	15



ELEVEN

Again, in this context only, the COLOR command corresponds to the actual color, and not a color register number which contains the value of the color information.

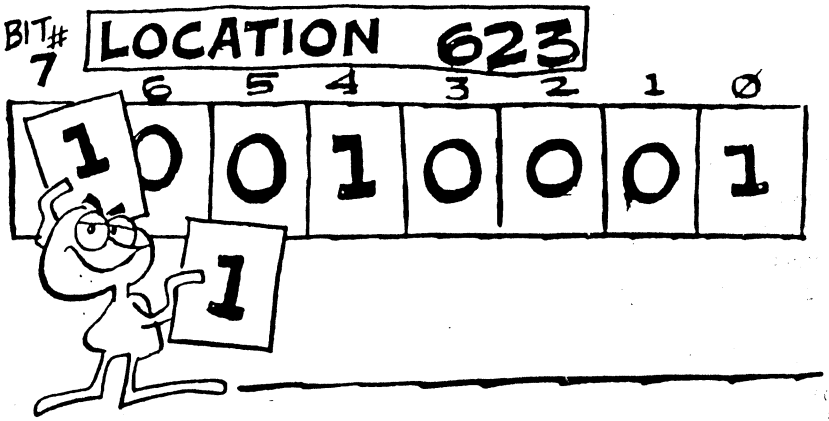
4) After selecting a color, PLOT and DRAWTO normally. The acceptable boundaries within which you must PLOT and DRAWTO are the same as for Mode 9.

The next program, ELEVEN, demonstrates the use of 16 colors at the same time. This mode is useful for the most colorful displays available on the ATARI utilizing a single brightness level.

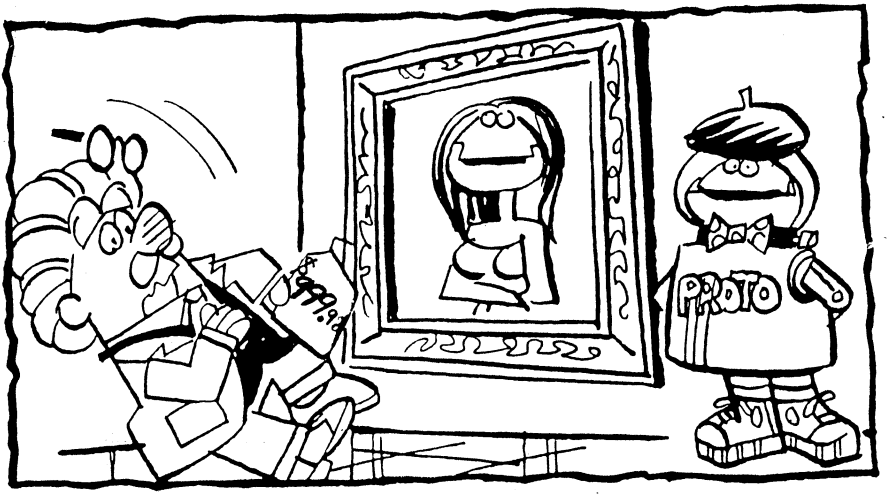
To tell the truth, this mode seems to be the least useful of the three. Mode 9 gives you extended shading, and Mode 10 gives you almost as many colors, while keeping the valuable feature of color indirection. All that Mode 11 gives you is all 16 colors. Since many of these colors are so close to each other that it is hard to tell them apart, the advantage of having all 16 available at the same time is questionable. But it is nice to know that you have a rainbow available when you need it.

WHY IS MODE 8 LIKE SYBIL?

Give up? The answer is multiple personalities. The new modes 9,10, and 11 are really just variations on graphics mode 8. So a lot of the nitty-gritty details will be the same, such as the amount of memory required to map the screen (about 8k). The computer tells which variety of mode 8 display (regular or 9-11) is the flavor of the nanosecond by checking bits 6 and 7 of the PRIORity register, which you can read at location 623. If the value at that location is greater than 64, then one of the GTIA bits is set (to be exact, a value between 64 and 128 means mode 9, between 128 and 192 gives mode 10, and 192 and over is used for mode 11). That's how the above test for the presence of the chip works. Mode 0 shares enough characteristics with mode 8 so that setting the GTIA bits confuses the mode 0 display!



When the GTIA bits are set, the computer interprets display memory differently. In mode 8, each memory location keeps track of 8 consecutive points of light on the screen. Without boring you with too much binary arithmetic, let's just say that each byte of memory can hold a value from 0 to 255, which corresponds to the number of combinations of lit and unlit dots you can make with 8 dots. But in GTIA mode, not only can each dot be turned on or off, but also must contain information about which one of 16 hues or brightness levels should be displayed. With each byte of screen memory doing so much more work, something has got to give. The result is that GTIA modes require some trade-offs for the sake of added color.



THE PRICE YOU PAY FOR ART

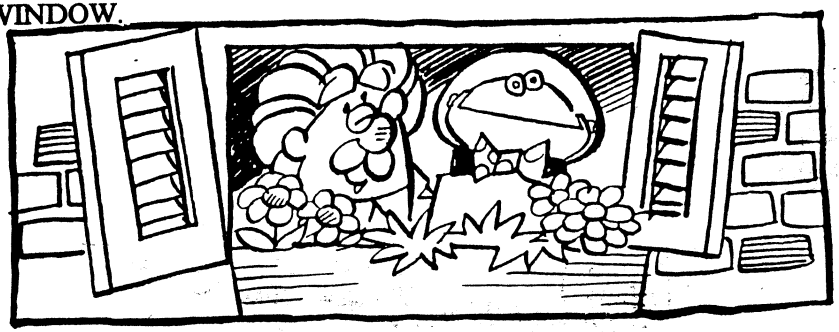
The first of these trade-offs is in the number of dots per row that can be displayed. In mode 8, 320 separate dots can be displayed in each row, giving you nice little points of light to work with. In modes 9,10, and 11 you need extra color information for each dot, so that you can only get 80 dots per row given the same amount of display memory as mode 8. The result is cute little rectangles that are about 4 times as wide as they are high. The resolution of these modes is somewhere between modes 5 and 7--medium resolution.

The next trade-off is loss of color indirection in modes 9 and 11. For those of you who wonder how you can lose something you didn't even know you had, I'll explain that term very simply. Color indirection describes ATARI's system for keeping track of the color of each dot. In most memory-mapped graphics, each unit of screen memory keeps track of the color of a particular dot on the screen. If the number in memory is 0, the dot might be black, if it's 1 it might be white, if it's 2 it might be blue, etc. Those of you who are on your toes know that in your ATARI computer, what screen memory keeps track of is the number of a color register (that's the number specified by the BASIC COLOR statement). When CTIA/GTIA wants to know what color to throw on the screen, it consults the register. The result is that if you color several squares on the screen with the contents of register 0, you can change the color of all of those squares at once just by changing the color value of that register (in BASIC, you would use the SETCOLOR command).

WINDOW

When it comes to the GTIA modes, there is just one small problem—we don't have 16 color registers available! There are 5 regular color registers plus an additional 4 for Player-Missile graphics, giving us a grand total of 9. The only way to get the additional colors is to go back to the system used in most other personal computers—having the color value of each dot on the screen directly represent the color or shade of brightness of the dot, rather than indicating the color register to look into to find that value. Modes 9 and 11 use this system, allowing 16 shades of one color, or sixteen different colors at one brightness level. The drawback with this system comes when you want to change the color of some dots on the screen. You are going to have to redraw every single one of them in the new color. Quelle bummer! But for those programs requiring the color-change tricks made possible by color indirection, ATARI has a third alternative. As mentioned previously, Mode 10 uses the 5 regular and 4 P-M registers to give you a choice of 9 different colors, each of which can be selected from the regular ATARI palette of 16 colors at 8 levels of brightness. All this, and indirection too!

Another option that you lose with the GTIA modes is that of choosing a text window at the bottom of the screen. The reason for that should be obvious after trying the little test of POKEing 623 with a 64. Enabling the GTIA modes confuses the mode 0 display as well, so that even if you created a text window, the display would be unreadable. There is a way to mix GTIA map-mode graphics with text, however. First, you create a Graphics 8 screen with a text window. Then you POKE 623 with a value above 64, as explained above. Now you have a GTIA mode with a confused, unreadable window. The next step requires a Display List Interrupt (DLI), a little machine-language program that steps in before the whole screen display is completed, and changes the value of one or more graphics registers. This interrupt allows you to change the contents of the PRIORity register for just a few lines of the screen, so that you can disable the GTIA mode just for your text window. A sample program showing how to do this is given in the next program, WINDOW.

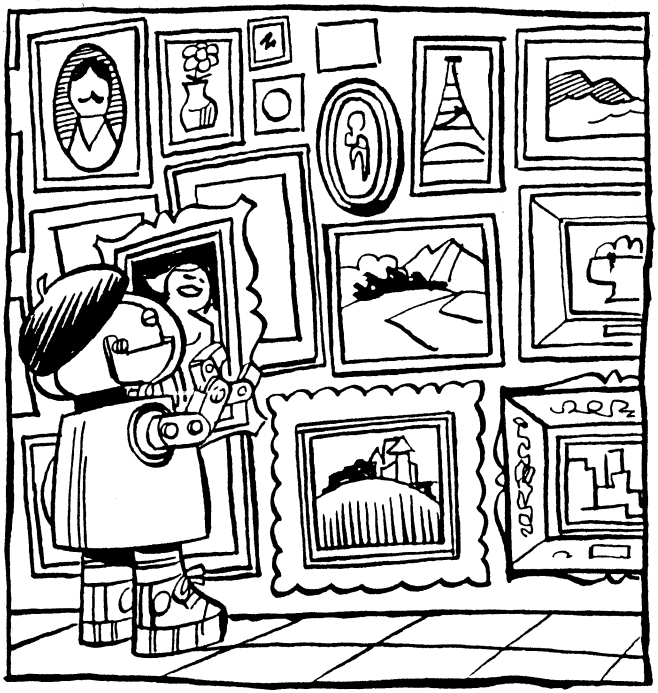


WHAT ABOUT PLAYER-MISSILES?

There are other, even more subtle differences between the GTIA modes and the normal playfield map-modes (3-8). You beginners can scan this section, but don't worry if it doesn't make too much sense at this point. If you get into heavy programming later on, you can always refer back to it.

One of these differences concerns Player-Missile Graphics. As you know from my Player-Missile Graphics Tutorial#5, there are memory locations called collision registers that keep tabs of the position of the Players and Missiles and of the normal (playfield) graphics dots and can let you know when two or more of these overlap. In that way, you can check, for example, whether a player has bumped into a wall, so you know that you can no longer move him in the same direction. Well, the way that the collision registers work involves color indirection; they tell whether an object which gets its color from one register has been in contact with an object that gets its color from another register. As I told you above, however, in GR. 9 and 11 the playfield dots do not get their colors from any register. Therefore, the collision registers cannot tell if the players or missiles have bumped into any of those colored dots. This significantly limits the power of Player-Missile graphics in those modes. Since mode 10 does use color indirection, you can detect most collisions. But you will notice that some of the playfield dots get their colors from the Player-Missile color registers. Therefore, collisions between these dots and Players will be treated the same as collisions between two players.

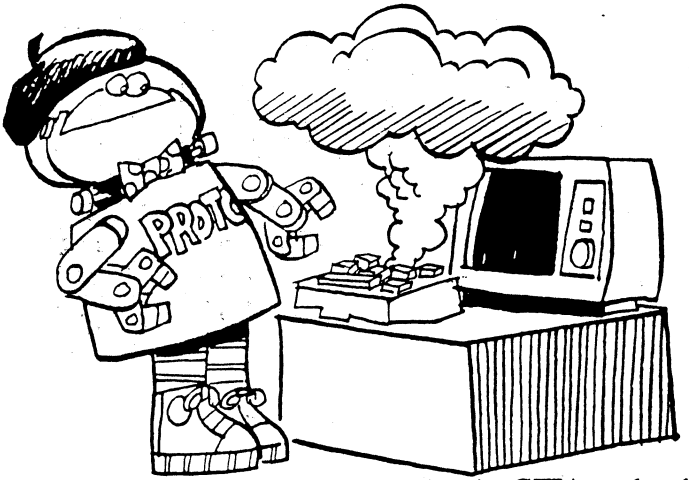
An even more obscure difference is in the treatment of the PUT statement. Since ATARI Input/Output operations are designed to work the same for any kind of device, you should be able to use the I/O commands PRINT#6 and PUT#6 in map modes, as well as use the COLOR, PLOT sequence of commands. Give it a try. Get into mode 3 for example, and PRINT #6;"ABC". You will see the dots corresponding to the screen data for those letters appear. Likewise, you could PUT#6;1, and see that dot appear. But in GTIA modes, the system has trouble keeping track of the last dot printed, and the number of dots per line, so that if you try outputting say 1000 characters with a PUT statement in a loop, you would get a "cursor out of range error". It's not likely that this is going to come up in your programming, but if it ever does, remember where you heard it.



"I don't know where to PUT this!"

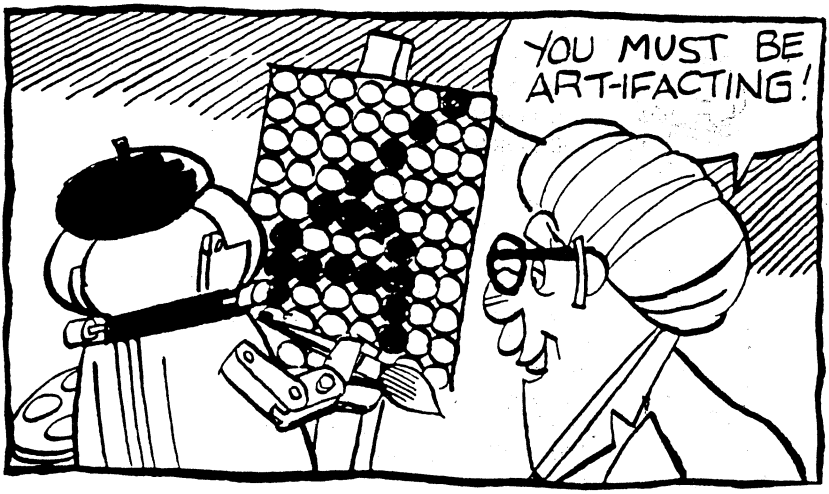
SOFTWARE COMPATABILITY BETWEEN CTIA AND GTIA MACHINES

Many people have been concerned lately because they have heard that some software that works on CTIA machines will not work on GTIA machines. Actually, many of the problems which have been the subject of such rumors have had nothing to do with the substitution of the GTIA chip for the CTIA chip. Most of them were caused because at about the same time ATARI started shipping machines with the GTIA, they also changed over to the revision B 10K ROM Operating System. This new OS fixed a few minor problems with the old one, but in doing so changed some of the internal routines around. Some software had been using those routines, even though ATARI told developers not to do it (bad software!). The result was that when the software came looking, the routines had moved, and the program crashed. Most of those software developers learned their lessons, and have fixed the software to be compatible with the new ROMS. The ATARI 1200XL has similar problems since an additional 4K of OS was added.



Although theoretically any program not using the GTIA modes should run the same on the CTIA machines as the GTIA machines, some software producers have gone a long way to screw up. They have set the GTIA bits in the PRIORity register for no good reason, making the program graphics go wild in GTIA machines. An example of this was early versions of Crush, Crumble and Chomp. This too has been fixed. The moral of the story is don't set bits for no good reason, particularly if you don't know what they do.

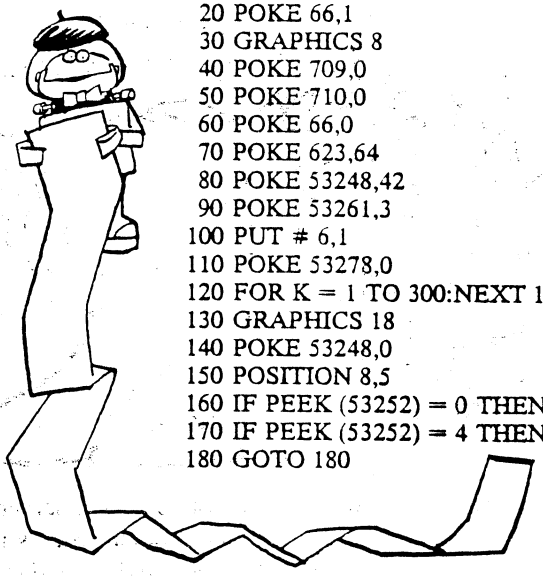
There is one real difference in the way that GTIA machines and CTIA machines display graphics, however, and that is in the area of color artifacts. No, we're not talking about archaeological ruins. Color artifacts is the term used to describe the fact that each dot of light in the highest resolution modes (8 and 0) is a slightly different color than its horizontal neighbor. Therefore, if you draw vertical lines using only dots positioned in odd-numbered columns, you will get a different color than if you draw those lines using only even-numbered dots. If you draw those lines so that odd and even numbered dots are lit next to one another, you will get the normal color, as we saw in the program, DEMO.



Many programs use artifacting to put more colors on the screen when using the high-resolution modes. Here is where the problem comes in. The display that the GTIA puts on the TV is shifted over one dot relative to that put on the screen by the CTIA. In other words, the color generated by odd columns of dots with the CTIA is the same color generated by even columns of dots with the GTIA, and vice versa. This is the reason that the maze in the original Jawbreaker, for example, appears to be blue when displayed by CTIA machines, but bright green when displayed by GTIA machines. In some programs the difference in color does not seriously affect the display, but in others it can turn the color scheme into a nightmare. If the program could tell what kind of machine it was running on, then it could shift the color rows over if necessary. But as great an authority as De Re ATARI has said that "there is currently no way for a program to determine whether or not a GTIA is present in a system". What's a programmer to do? Why, ask the Professor of course.

WINDOW

If you remember back to the discussion of Player-Missile Graphics, you will notice that I said that collision detection was different between systems with a CTIA and a GTIA. Therefore, if you try to enable the GTIA mode, and then set up a collision, the collision will be reported if you have a CTIA machine, but it will not be reported if you have a GTIA machine. This is a definite drawback for GTIA games as the collision registers will not work. The following routine shows exactly how this is done. This routine could be used at the beginning of your programs to test which chip the user has and branch accordingly to the correct version. Here is the code:



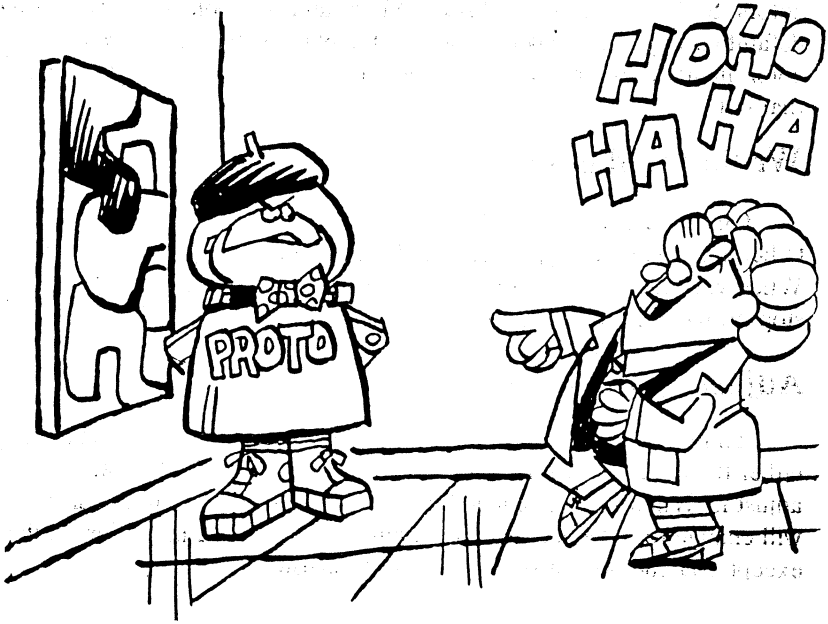
```
10 REM Test for CTIA/GTIA
20 POKE 66,1
30 GRAPHICS 8
40 POKE 709,0
50 POKE 710,0
60 POKE 66,0
70 POKE 623,64
80 POKE 53248,42
90 POKE 53261,3
100 PUT # 6,1
110 POKE 53278,0
120 FOR K = 1 TO 300:NEXT 18K
130 GRAPHICS 18
140 POKE 53248,0
150 POSITION 8,5
160 IF PEEK (53252) = 0 THEN ? # 6;"GTIA"
170 IF PEEK (53252) = 4 THEN ? # 6;"CTIA"
180 GOTO 180
```

This determines what kind of a machine you are using and can easily be modified to branch to different sections of your programs by altering lines 160 and 170. The section ARTIFACT in the DEMO program shows how to use this information in a program. It draws squares in the regular mode 8 color, and the two artifacted colors. Although these colors vary from set to set, one of them is generally green, and the other blue. Without using a software test to adjust the program according to the type of chip, the block on the right would be blue on a GTIA machine and the block on the left would be green, while on a CTIA machine, the colors would be reversed. With this test, however, the program can draw the lines in alternate even, or odd numbered columns as is appropriate, so that the green block is on the left, regardless of the type of machine used.

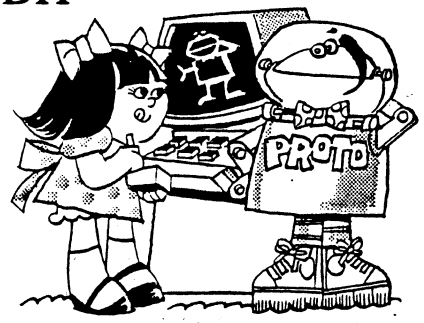
WINDOW

Advanced programmers should also be aware of a subtle difference in the position of Players relative to normal (playfield) graphics (beginners are excused here, and can take a break). Every dot in each player is twice as wide as the narrowest dot in high-resolution map mode. This means that every Player is going to start in either an even numbered column or an odd numbered column. Because of the shift of the GTIA display relative to the CTIA display, Players start in even numbered columns on a GTIA machine, but in odd numbered columns on a CTIA machines. Since mode 0 letters all start in even numbered columns, there is no way you can get a Player to exactly overlap the space occupied by, say, the mode 0 cursor if you are working with a GTIA machine.

In GTIA modes, you can get Players to overlap mode 0 characters precisely. Just keep this in mind if you are trying to set up a stationary display using both letters and Players–the display may look a little off on a CTIA machine. Check it out to make sure. You can correct for positioning of Players after testing for a CTIA with the GTIATEST routine on the previous page. To complicate matters further, the four-color ANTIC modes 4 and 5 are also shifted one dot space on CTIA machines. Therefore, if you are trying to mix mode 0 lines with ANTIC modes 4 and 5, make sure that the positioning doesn't look too funny on CTIA machines.



GEDIT



THE GTIA EDITOR

Now here's your chance to create great medium resolution drawings and SAVE them to disk or tape. LOADing them later on is just as easy. When the program GEDIT begins, you'll notice a white screen with several colored blocks at the bottom. In the center of the screen are four flashing little marks forming a rectangular shape. The colored blocks are your color choices. The flashing little marks are the boundaries of your drawing cursor. Plug a joystick into port one and let's get started:

Picking a Color

Press the letter 'C' to select a color. A dark line will appear under the color presently being used. To change the color, move your joystick right or left. If you wish to change the color, move your joystick up or down. You can adjust to any shade or color available. Be forewarned though, that when you adjust the colors, SAVE a picture, and reLOAD it later, the colors will be those of the default variety you see when you first RUN the GEDIT program. To select a color, lightly press the joystick trigger button. You're now ready to draw using that color.

Writing to the Screen

Once you've chosen a color, position the flashing cursor where you want it to be. Simply move the joystick up, down, or diagonally to position the cursor. When you wish to draw, press the trigger button down and hold it down. To stop drawing, release the trigger.

Adjusting the Cursor Size

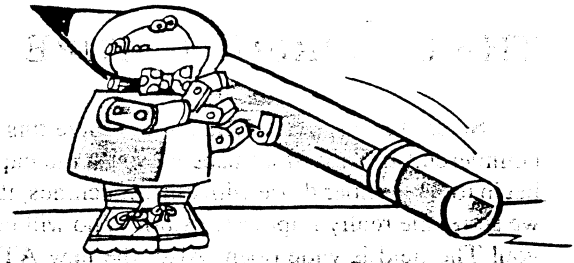
To make the drawing cursor taller or shorter, press the CTRL key and either the up and down arrow. A little sound will chime for each size as you adjust it. When you've reached the maximum or the minimum size, the tone will change to a deeper tone. To adjust the width, the procedure is the same except you should use the left and right arrows.

Clearing the Screen

To erase the entire screen, press the SHIFT key and the CLEAR key simultaneously. The entire screen should clear. Be sure to SAVE any work you want to keep before using this option.

Erasing

Press the 'E' key to change the cursor color to the white background color. You may then erase as little or as much as you desire by pressing the trigger button when you wish to erase and releasing it when you don't want to erase. You may move the cursor and adjust its size in the usual manner. To start drawing again, select a new color by pressing 'C'.



Saving a Drawing

For the disk version, press the 'S' key to SAVE a drawing. A small tone will sound at which time you should press 'S' again to SAVE. Press any other key to abort the save. Your drawing will be SAVEd under the name TEMP.DRW.

CAUTION!!

To save more than one drawing on the same disk, it will be necessary to rename the old drawing to prevent it from being overwritten.

For the tape version, position a blank tape before pressing the 'S' key. The internal speaker will sound indicating you should press PLAY on your program recorder. Press the 'S' key again to SAVE to tape. No name will be assigned. You must keep track of the starting position of the tape so you can reLOAD it later on.

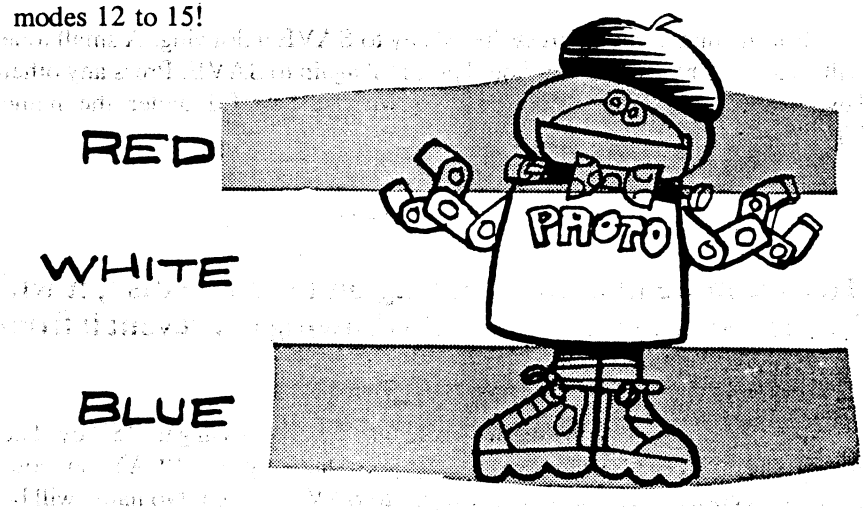
Loading a Drawing

To LOAD a drawing from disk, insert the disk with the drawing TEMP.DRW into your disk drive. Press the 'L' key and a small tone will sound. Press the 'L' key again to LOAD the drawing. Pressing any other key will abort the LOAD. Your drawing will then appear on your screen where you can continue to work on it.

To LOAD a drawing from tape, position the tape with the desired drawing on it. Press the 'L' key to sound the internal speaker prompt. Press PLAY on your program recorder and press the 'L' key again. Your picture should now appear on the screen.

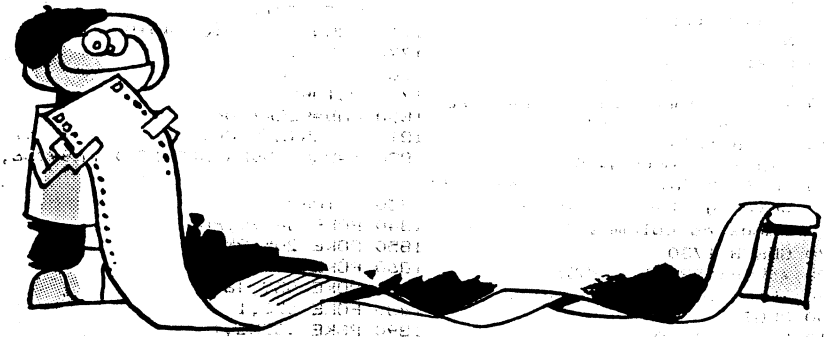
THE COLORFUL FUTURE

Since the early ATARIs did not have this chip, we have yet to see any commercial software that uses its awesome capabilities. But as new owners begin to outnumber those with older machines, it is just a matter of time before we see some really impressive stuff. Who will be the first? It might as well be you! The field is wide open. Also, the new ATARI 1200 has GRAPHICS modes 12 to 15!



THE GTIA PROGRAM LISTINGS

This section contains selected program listings of the GTIA software for you to study. The assembly code has been stripped leaving only the BASIC code. If you desire to see the assembly code of the program currently in memory, press the BREAK key, type LIST, and press RETURN. The code should scroll down your screen. To stop and start the scrolling, press the CTRL key and number 1 simultaneously.



GTIATEST

```
1000 REM GTIATEST
1010 M=6:POKE 82,M:? "?"
1020 ? ""
1030 ? "!        GTIA CHIP TEST        !"
1040 ? ""
1050 ? "      AS THE TONE SOUNDS"
1060 ? ""
1070 ? "!IF YOUR SCREEN TURNS BLACK!"
1080 ? "!YOUR COMPUTER HAS THE GTIA!"
1090 ? "!        CHIP INSTALLED!     !"
1100 ? ""
1110 ?
1120 ? ""
1130 ? "!IF THE SCREEN REMAINS BLUE!"
1140 ? "!THEN YOU HAVE THE OLD CHIP!"
1150 ? "! AND IT MUST BE REPLACED!  !"
1160 ? ""
1170 POSITION M,21
1180 ? "PUSH ANY KEY TO CONTINUE.. ";
1190 OPEN #1,4,0,"K:":GET #1,A
1200 POKE 82,2
1210 GRAPHICS 9
1220 FOR W=1 TO 200
1230 SOUND 0,40,10,10
1240 NEXT W
1250 SOUND 0,0,0,0
1270 GRAPHICS 10:FOR Z=704 TO 712
     :READ R:POKE Z,R:NEXT Z
1280 DATA 0,26,42,58,74,90,106,122,
     138,154
1290 FOR X=1 TO 8:COLOR X:POKE 765,X
1300 PLOT X*4+5,0:DRAWTO X*4+5,159
     :PLOT X*4+1,159:POSITION X*4+1,0
     :XIO 18,#6,0,0,"S:"
1310 NEXT X
1320 FOR X=8 TO 15:COLOR 16-X
     :POKE 765,16-X
1330 PLOT X*4+5,0:DRAWTO X*4+5,159
     :PLOT X*4+1,159:POSITION X*4+1,0
     :XIO 18,#6,0,0,"S:"
1340 NEXT X
1350 COLOR 0:PLOT 65,159:DRAWTO 0,159
1360 POKE 764,255
1370 FOR I=1 TO 40
1380 FOR X=1 TO 8:Z=PEEK(704+X)
     :Z=Z+16:IF Z>255 THEN Z=26
1390 SOUND 0,Z,10,10
1400 K=PEEK(764)
1410 POKE 704+X,Z:NEXT X
     :FOR Y=1 TO 5:NEXT Y
1420 NEXT I
1430 SOUND 0,0,0,0
1440 GRAPHICS 2+16:POSITION 0,4
1450 ? #6;"  RETURN TO REPEAT"
1460 ? #6;"                  "
1470 ? #6;"   PUSH SPACE BAR "
1480 ? #6;"    TO RUN  DEMO  "
1490 POKE 764,255
1500 K=PEEK(764):IF K=255 THEN 1500
1510 POKE 764,255:IF K=12 THEN RUN
1520 POKE 65,0:RUN "D:DEMO"
```

DEMO

```
1000 REM DEMO
1002 DIM WT$(57)
1020 DIM CM$(36)
1040 DIM BS$(112)
1060 DIM MP$(118)
1100 DIM TEXT$(320)
1110 GOTO 3010
1120 GOSUB 1800
1130 GRAPHICS 8
1140 GOSUB 1840
1150 FOR I=50 TO 100 STEP 2
1160 PLOT I,40
1170 DRAWTO I,100
1180 NEXT I
1190 P1=30:P2=80:ST=6
1220 GOSUB 1620
1230 TEXT$=" Hmm..this looks like
     artifacted color"
1240 GOSUB 1730
1250 X=USR(ADR(WT$),255)
1260 TEXT$=" This stuff comes from
     drawing lines   only in even
     numbered columns "
1270 GOSUB 1730
1280 X=USR(ADR(WT$),255)
1290 FOR I=221 TO 271 STEP 2
1300 PLOT I,40
1310 DRAWTO I,100
1320 NEXT I
1330 P1=80
1340 P2=160
1350 ST=6
1360 GOSUB 1620
1370 TEXT$=" OH NO..more of this
     artifacted color stuff"
1380 GOSUB 1730
1390 X=USR(ADR(WT$),255)
1400 TEXT$=" This is the kind
     that comes when you draw in
     ODD numbered columns"
1410 GOSUB 1730
1420 X=USR(ADR(WT$),255)
1430 FOR I=135 TO 185
1440 PLOT I,40
1450 DRAWTO I,100
1460 NEXT I
1470 P1=160
1480 P2=130
1490 ST=-6
1500 GOSUB 1620
1510 TEXT$=" AHHHH...Now THAT'S
     more like it.."
1520 GOSUB 1730
1530 X=USR(ADR(WT$),255)
1540 TEXT$=" True color from BOTH
     odd and even      columns!"
1550 GOSUB 1730
1560 X=USR(ADR(WT$),255)
1570 FOR I=4 TO 256 STEP 16
1580 POKE 710,I
1590 X=USR(ADR(WT$),100)
1600 NEXT I
1610 GOTO 3010
1620 FOR HZ=P1 TO P2 STEP ST
1630 FOR BEAST=0 TO 2
1640 X=USR(ADR(MP$),0,BEAST,80,HZ)
1650 IF BEAST=1 THEN 1680
1660 SOUND 0,80,10,10
1670 X=USR(ADR(WT$),4)
1680 SOUND 0,0,0,0
1690 NEXT BEAST
1700 NEXT HZ
1710 X=USR(ADR(MP$),0,1,80,HZ-6)
1720 RETURN
1730 ? "}"
1740 FOR I=1 TO LEN(TEXT$)
1750 ? TEXT$(I,I);
1760 SOUND 0,RND(0)*40+80,10,10
1770 NEXT I
1780 SOUND 0,0,0,0
1790 RETURN
1800 PMB=PEEK(106)-36
1810 X=USR(ADR(CM$),PMB*256,1024)
1820 X=USR(MOVER,ADR(BS$),PMB*256,
     112)
1830 RETURN
1840 POKE 54279,PMB
1850 POKE 205,PMB
1860 POKE 53277,1
1870 POKE 559,46
1880 POKE 623,1
1890 POKE 53256,1
1900 POKE 704,14
1910 COLOR 1
1920 POKE 752,1
1930 ? " "
1940 RETURN
1950 BAS=PEEK(106)-8
1960 X=USR(MOVER,57344,BAS*256,1024)
1970 X=USR(MOVER,ADR("
1980 GOSUB 1800
1990 GRAPHICS 1
2000 GOSUB 1840
2010 POKE 756,BAS
2020 POSITION 15,10
2030 ? #6;"33333"
2040 P1=40
2050 P2=160
2060 ST=6
2070 GOSUB 1620
2080 TEXT$=" What have we here?"
2090 GOSUB 1730
2100 X=USR(ADR(WT$),255)
2110 TEXT$=" Looks like Mode FIVE
     wavy lines...    pretty
     coarse resolution"
2120 GOSUB 1730
2130 X=USR(ADR(WT$),255)
2140 POSITION 8,10
2150 ? #6;"222222"
2160 P1=160
2170 P2=120
2180 ST=-6
2190 GOSUB 1620
2200 TEXT$=" This must be
     Mode SEVEN"
2210 GOSUB 1730
```

```
2220 X=USR(ADR(WT$),255)
2230 TEXT$=" That's a little better
     resolution"
2240 GOSUB 1730
2250 X=USR(ADR(WT$),255)
2260 POSITION 1,10
2270 ? #6;"010101"
2280 P1=120
2290 P2=70
2300 ST=-6
2310 GOSUB 1620
2320 TEXT$=" This is what the
     GTIA modes look like"
2330 GOSUB 1730
2340 X=USR(ADR(WT$),255)
2350 TEXT$=" Their dots are short
     and fat...
     kinda medium resolution"
2360 GOSUB 1730
2370 X=USR(ADR(WT$),1000)
2380 GOTO 3010
2390 GOSUB 1800
2400 GRAPHICS 7
2410 GOSUB 1840
2420 FOR I=1 TO 150
2430 COLOR INT(RND(0)*3)+1
2440 PLOT RND(0)*158,RND(0)*60
2450 NEXT I
2460 FOR I=1 TO 3
2470 FOR J=1 TO 20
2480 COLOR I
2490 PLOT I*20+J+10,10
2500 DRAWTO I*20+J+10,60
2510 NEXT J
2520 NEXT I
2530 P1=40
2540 P2=90
2550 ST=6
2560 GOSUB 1620
2570 TEXT$=" Here we have colors
     from three
     different registers"
2580 GOSUB 1730
2590 X=USR(ADR(WT$),600)
2600 TEXT$=" Change the register,
     and you change
     everything that's that color"
2610 GOSUB 1730
2620 X=USR(ADR(WT$),600)
2630 FOR I=708 TO 710
2640 FOR J=2 TO 255
2650 POKE I,J
2660 NEXT J
2670 NEXT I
2680 POKE 708,132
2690 POKE 709,132
2700 POKE 710,132
2710 FOR J=1 TO 30
2720 FOR I=708 TO 710
2730 POKE I,14
2740 X=USR(ADR(WT$),10)
2750 POKE I,132
2760 NEXT I
2770 NEXT J
2780 POKE 708,40
2790 POKE 709,202
2800 POKE 710,148
2810 COLOR 125
2820 PLOT 1,1
2830 FOR I=1 TO 3
2840 FOR J=1 TO 20
2850 COLOR I
2860 PLOT I*20+J+10,10
2870 DRAWTO I*20+J+10,60
2880 NEXT J
2890 NEXT I
2900 TEXT$=" Its much slower to
     change colors by redrawing"
2910 GOSUB 1730
2920 X=USR(ADR(WT$),600)
2930 FOR I=1 TO 3
2940 FOR J=1 TO 20
2950 COLOR I-1+3*(I=1)
2960 PLOT I*20+J+10,10
2970 DRAWTO I*20+J+10,60
2980 NEXT J
2990 NEXT I
3000 X=USR(ADR(WT$),1000)
3010 POKE 53248,0
3020 GRAPHICS 17
3030 ? #6
3040 ? #6;"    WHICH EXAMPLE?"
3050 ? #6
3060 ? #6
3070 ? #6;"    1) RESOLUTION"
3080 ? #6
3090 ? #6;"    2) INDIRECTION"
3100 ? #6
3110 ? #6;"    3) ARTIFACTING"
3120 ? #6
3130 ? #6
3140 ? #6;"    4) QUIT"
3150 CLOSE #1:OPEN #1,4,0,"K:"
3160 GET #1,A
3170 ON A-48 GOTO 1950,2390,1120,9000
3180 GOTO 3160
9000 RUN "D:PICTURE"
```

PICTURE

```
1000 REM PICTURE
1010 REM ; by STAN GILBERT
1020 REM ; (C)Copyright 4/24/83
2000 REM
2010 DIM WT$(57):REM __TDELAY.OBJ
2050 REM
2060 DIM FM$(54):REM __FILMEM.OBJ
2100 REM
2110 DIM MM$(49):REM __MOVMEM.OBJ
2150 REM
2160 DIM CC$(37):REM __CIOCTL.OBJ
2190 REM
2200 DIM SP$(48):REM __SETPMS.OBJ
2240 REM
2250 DIM MP$(118):REM __MOVPMS.OBJ
2320 REM
2330 DIM SS$(95):REM __SOUNDS.OBJ
4000 DIM F$(20),M$(80)
4010 GRAPHICS 0:POKE 752,1
4020 ? ""
4030 ? "!                PICTURE MENU
4040 ? ""
4050 ? "
4060 FOR I=4 TO 22
4070 POSITION 2,I:? "!"
4080 POSITION 37,I:? "!"
4090 NEXT I
4100 ? "";:POSITION 5,3
4102 RESTORE
4110 FOR I=8 TO 16 STEP 2
4120 READ F$:POSITION 10,I
4130 ? (I/2)-3;". ";F$
4140 NEXT I
4150 POSITION 3,22
4160 ? " [illegible] Enter selection? ";
4180 POKE 764,255
4190 K=PEEK(764):IF K=255 THEN 4190
4200 POKE 764,255
4210 IF K=47 THEN 9010
4220 F=1:IF K=30 THEN F=2
4230 IF K=26 THEN F=3
4240 IF K=24 THEN F=4
4250 IF K=29 THEN F=5
4260 RESTORE :F$="D:"
4270 FOR I=1 TO F:READ M$:NEXT I
4280 F$(3)=M$:F$(LEN(F$)+1)=".DRW"
4290 IF F<>1 THEN 4299
4292 GRAPHICS 10
4293 RESTORE 4382
4294 FOR I=704 TO 712
4295 READ B:POKE I,B
4296 NEXT I
4299 IF F>1 THEN GRAPHICS 9
4300 SCA=PEEK(88)+PEEK(89)*256
4310 CLOSE #1:OPEN #1,4,0,F$
4320 X=USR(ADR(CC$),1,7,SCA,7680)
4330 POKE 764,255
4340 K=PEEK(764):IF K=255 THEN 4340
4350 POKE 764,255:GOTO 4010
4380 GOTO 4330
4370 DATA EDUSOFT,KOALA,LION,
     MONALISA
4380 DATA LINCOLN
4382 DATA 12,0,29,66,71
4384 DATA 123,156,214,234
9000 REM __PROGRAM TERMINATION
9010 FOR I=53248 TO 53255
9020 POKE I,0:NEXT I
9030 FOR I=0 TO 3
9040 SOUND I,0,0,0:NEXT I
9042 REM POKE 106:P106
9050 GRAPHICS 2+16:POSITION 0,4
9060 ? #6;"  RETURN TO REPEAT"
9070 ? #6
9080 ? #6;"   PUSH SPACE BAR "
9090 ? #6;"    TO RUN NINE   "
9100 POKE 764,255
9110 Q=PEEK(764):IF Q=255 THEN 9110
9120 POKE 764,255:IF Q=12 THEN RUN
9130 RUN "D:NINE"
```

NINE

```
1000 REM NINE
1010 REM __Assembly routines
1020 REM
1030 DIM WT$(57):REM __TDELAY.OBJ
1070 REM
1080 DIM FM$(54):REM __FILMEM.OBJ
1120 REM
1130 DIM MM$(49):REM __MOVMEM.OBJ
1170 REM
1180 DIM SS$(95):REM __SOUNDS.OBJ
1240 REM
1250 DIM CC$(37):REM __CIOCTL.OBJ
1280 REM
1290 DIM SP$(48):REM __SETPMS.OBJ
1330 REM
1340 DIM MP$(118):REM __MOVPMS.OBJ
1410 REM
1420 DIM BS$(136):REM __BEASTS.OBJ
1500 REM
1510 REM __Mainline
1520 REM
1530 P106=PEEK(106)
1540 PB=P106-4:PA=PB*256
1550 POKE 106,PB-12
1560 GRAPHICS 9
1570 X=USR(ADR(SP$),PB)
1580 X=USR(ADR(FM$),PA,0,1024)
1590 X=USR(ADR(MM$),ADR(BS$),PA,136)
1600 POKE 623,33+64
1610 POKE 704,58
1620 U=USR(ADR(MP$),0,1,63,55)
1630 REM __ [illegible]
1640 FOR I=80 TO 96
1650 COLOR I:PLOT 0,I:DRAWTO 66,I
1660 READ B:COLOR B:J=112-(I-80)
1670 PLOT 0,J:DRAWTO 66,J
1680 NEXT I
```

```
1690 REM __ DRAW RINGS
1700 RESTORE
1710 FOR I=64 TO 96
1720 READ B:COLOR B:J=128-(I-64)
1730 PLOT 25,I:DRAWTO 26,I
1740 PLOT 25,J:DRAWTO 26,J
1750 PLOT 44,I:DRAWTO 45,I
1760 PLOT 44,J:DRAWTO 45,J
1770 NEXT I
1780 REM __ DRAW VALVE HOUSING
1790 FOR I=27 TO 43
1800 READ A,B:K=1
1810 FOR J=0 TO 21 STEP 3
1820 IF J<9 OR J>12 THEN K=K+1
1830 IF J=9 THEN K=7
1840 IF J=12 THEN K=10
1850 IF J=15 THEN K=12
1860 COLOR K
1870 PLOT I,A+J:DRAWTO I,B-J
1880 NEXT J
1890 NEXT I
1900 REM __ DRAW HANDLE
1910 COLOR 2
1920 FOR I=33 TO 35
1930 IF I<>34 THEN 1950
1940 PLOT I,47:DRAWTO I,64
1950 PLOT I,48:DRAWTO I,64
1960 NEXT I
1970 REM
1980 COLOR 6
1990 FOR J=36 TO 38
2000 IF J<>37 THEN 2020
2010 PLOT J,47:DRAWTO I,64
2020 PLOT J,48:DRAWTO J,64
2030 NEXT J
2040 COLOR 3:PLOT 35,28:DRAWTO
     35,46
2050 COLOR 10:PLOT 36,28:DRAWTO
     36,47
2060 COLOR 6:PLOT 37,28:DRAWTO
     37,46
2070 FOR Z=27 TO 34
2080 COLOR Z+12
2090 PLOT 23,Z:DRAWTO 49,Z
2100 NEXT Z
2110 FOR H=55 TO 180 STEP 6
2120 FOR B=0 TO 2
2130 U=USR(ADR(MP$),0,B,63,H+2*B)
2140 SOUND 0,20+I,10,10
2150 U=USR(ADR(WT$),6)
2160 SOUND 0,0,0,0
2170 NEXT B:NEXT H
2180 U=USR(ADR(MP$),0,1,63,H+B)
2190 U=USR(ADR(WT$),50)
2200 FOR I=63 TO 112
2210 U=USR(ADR(MP$),0,1,I,H+B)
2220 SOUND 0,I,10,10
2230 NEXT I
2240 SOUND 0,0,0,0
2250 POKE 106,P106:POKE 53248,0
2260 GRAPHICS 2+16:POSITION 0,4
2270 ? #6;"  RETURN TO REPEAT"
2280 ? #6;"                    "
2290 ? #6;"    PUSH SPACE BAR "
2300 ? #6;"    TO RUN  STICKS "
2310 POKE 764,255
2320 K=PEEK(764):IF K=255 THEN 2320
2330 POKE 764,255:IF K=12 THEN RUN
2340 RUN "D:STICKS"
7000 DATA 1,1,2,2,2,3,3,3,4,4,5,5
7010 DATA 6,6,7,7,8,8,9,9,10,10,11,11
7020 DATA 12,12,13,13,13,13,13,13,13
7030 DATA 77,115,77,115,74,116,70,119
7040 DATA 67,123,65,124,64,125,64,126
7050 DATA 63,127,63,126,64,126,64,125
7060 DATA 65,124,67,123
7070 DATA 70,119,74,116,77,115,77,115
```

STICKS

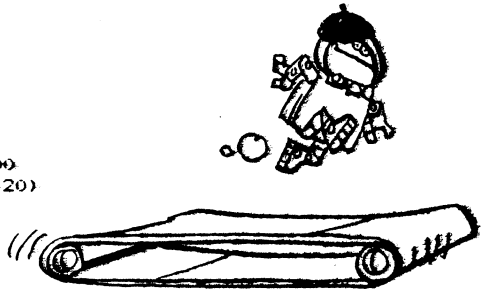
```
1000 REM STICKS
1010 GRAPHICS 10
1020 POKE 704,0
1030 FOR K=0 TO 2
1040 RESTORE
1050 FOR I=707 TO 712
1060 READ B:POKE I,B+K*32
1070 NEXT I
1080 DATA 2,4,6,8,6,4
1090 FOR I=2 TO 56 STEP 18
1100 FOR J=3 TO 18 STEP 3
1110 COLOR J/3+2
1120 PLOT I+J+K,10
1130 DRAWTO I+J+K,180
1140 NEXT J
1150 NEXT I
1160 A=PEEK(712):I=I-1
1170 FOR J=712 TO 700 STEP -1
1180 B=PEEK(J-1):POKE J,B
1190 NEXT J
1200 POKE 707,A
1210 IF I>1 THEN 1160
1220 NEXT K
1230 GRAPHICS 2+16:POSITION 0,4
1240 ? #6;"  RETURN TO REPEAT"
1250 ? #6;"                    "
1260 ? #6;"    PUSH SPACE BAR "
1270 ? #6;"   TO RUN  TREDMILL"
1280 POKE 764,255
1290 K=PEEK(764):IF K=255 THEN 1290
1300 POKE 764,255:IF K=12 THEN RUN
1310 RUN "D:TREDMILL"
```

TREDMILL

```
1000 REM TREDMILL
2000 REM __Assembly routines
2010 REM
2020 DIM WT$(57):REM __TDELAY.OBJ
2060 REM
2070 DIM FM$(54):REM __FILMEM.OBJ
2110 REM
2120 DIM MM$(49):REM __MOVMEM.OBJ
2160 REM
2170 DIM SS$(95):REM __SOUNDS.OBJ
2230 REM
2240 DIM CC$(37):REM __CIOCTL.OBJ
2270 REM
2280 DIM SP$(48):REM __SETPMS.OBJ
2320 REM
2330 DIM MP$(118):REM __MOVPMS.OBJ
2400 REM
2410 DIM BS$(136):REM __BEASTS.OBJ
4000 REM
4010 REM __Mainline code
4020 P106=PEEK(106)
4030 PB=P106-4:PA=PB*256
4040 POKE 106,P106-16
4050 X=USR(ADR(FM$),PA,0,1023)
4060 GRAPHICS 10
4070 X=USR(ADR(SP$),PB)
4080 POKE 623,46+128:POKE 53257,1
4090 X=USR(ADR(MM$),ADR(BS$),PA,
     136)
4100 FOR I=704 TO 712
4110 READ B:POKE I,B
4120 NEXT I
4130 DATA 0,10,54,2,4,6,8,6,4
4140 FOR I=2 TO 72 STEP 6
4150 FOR J=1 TO 6
4160 COLOR J+2
4170 PLOT I+J+K,180
4180 DRAWTO I+J+K,182
4190 NEXT J
4200 NEXT I
4210 FOR BS=0 TO 2
4212 IF PEEK(764)=47 THEN 9000
4220 X=USR(ADR(MP$),1,BS,97,120)
4230 GOSUB 4260
4240 NEXT BS
4250 GOTO 4210
4260 A=PEEK(712)
4270 FOR I=712 TO 708 STEP -1
4280 B=PEEK(I-1):POKE I,B
4290 NEXT I:POKE 707,A
4300 RETURN
9000 POKE 106,P106
9010 POKE 53249,0
9020 GRAPHICS 2+16:POSITION 0,4
9030 ? #6;"  RETURN TO REPEAT"
9040 ? #6;"                  "
9050 ? #6;"   PUSH SPACE BAR "
9060 ? #6;"  TO RUN  CYLINDER"
9070 POKE 764,255
9080 K=PEEK(764):IF K=255 THEN 9080
9090 POKE 764,255:IF K=12 THEN RUN
9100 RUN "D:CYLINDER"
```

CYLINDER

```
1000 REM CYLINDER
1010 DEG
1020 GRAPHICS 10
1030 FOR I=0 TO 7
1040 POKE 705+I,128+2
1050 NEXT I
1060 POKE 705,136
1070 FOR ANG=180 TO 360+180 STEP 6
1080 X=35+8*COS(ANG)
1090 Y=16+8*SIN(ANG)
1100 COLOR (ANG-180)/45+1
1110 PLOT X,Y:DRAWTO X,160+Y
1120 REM COLOR 1:PLOT X,Y
1130 NEXT ANG
1140 POKE 764,255
1150 A=PEEK(705)
1160 FOR I=705 TO 711
1170 IF PEEK(764)=47 THEN 1220
1180 POKE I,PEEK(I+1)
1190 NEXT I
1200 POKE 712,A
1210 GOTO 1150
1220 GRAPHICS 2+16:POSITION 0,4
1230 ? #6;"  RETURN TO REPEAT"
1240 ? #6;"                  "
1250 ? #6;"   PUSH SPACE BAR "
1260 ? #6;"  TO RUN  BLAKHOLE"
1270 POKE 764,255
1280 K=PEEK(764):IF K=255 THEN 1280
1290 POKE 764,255:IF K=12 THEN RUN
1300 RUN "D:BLAKHOLE"
```



BLAKHOLE

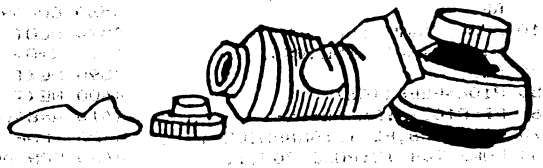
```
1000 REM BLAKHOLE
2000 REM Assembly routines
2010 REM
2020 DIM WT$(57):REM __TDELAY.OBJ
2060 REM
2070 DIM FM$(54):REM __FILMEM.OBJ
2110 REM
2120 DIM MM$(49):REM __MOVMEM.OBJ
2160 REM
2170 DIM SS$(95):REM __SOUNDS.OBJ
2230 REM
2240 DIM CC$(37):REM __CIOCTL.OBJ
2270 REM
2280 DIM SP$(48):REM __SETPMS.OBJ
2320 REM
2330 DIM MP$(118):REM __MOVPMS.OBJ
2400 REM
2410 DIM BS$(136):REM __BEASTS.OBJ
4000 REM
4010 REM __Mainline
4020 DIM A$(30),SINE(450)
4040 DEG
4050 P106=PEEK(106)
4060 PB=P106-4:PA=PB*256
4070 POKE 106,PB-12:GRAPHICS 1+16
4080 POKE 704,12:POKE 705,12
4090 POKE 752,1
4100 POSITION 2,10
4110 ? #6;"THE BLACK HOLE"
4120 POSITION 5,20:? #6;"STAND BY"
4130 FOR I=90 TO 0 STEP -1
4131 IF INT(I/10)<>I/10 THEN 4140
4132 POSITION 9,14:? #6;"
4133 POSITION 9,14:? #6;INT(I/10)
4136 U=USR(ADR(SS$),1)
4140 A=SIN(I):SINE(I)=A
4150 SINE(180-I)=A:SINE(I+180)=-A
4160 SINE(360-I)=-A:SINE(I+360)=A
4170 NEXT I
4180 GRAPHICS 10
4190 X=USR(ADR(SP$),PB)
4200 X=USR(ADR(FM$),PA,0,1024)
4210 X=USR(ADR(MM$),ADR(BS$),PA,
     136)
4220 POKE 623,46+128
4230 REM __Set color registers
4240 FOR I=704 TO 712
4250 READ B:POKE I,B
4260 NEXT I
4270 DATA 0,14,48,50,52,54,56,54,52
4280 REM GOTO 4640
4290 REM __Draw spiral
4300 TRAP 4430
4310 A=10:Q=2
4320 REM Q=3
4330 FOR ANG=0 TO 359 STEP 4
4340 X=A*SINE(ANG+90)+40
4350 Y=A*SINE(ANG)+96
4360 COLOR Q
4370 PLOT X,Y:DRAWTO X,Y+1
4380 A=A+0.025
4390 Q=Q+1:IF Q>8 THEN Q=2
4400 NEXT ANG
4410 GOTO 4320
4420 REM __Draw center of spiral
4430 FOR ANG=180 TO 359 STEP 2
4440 Q=2
4450 X=10*SINE(ANG+90)+40
4460 Y=10*SINE(ANG)+96
4470 COLOR 9:PLOT X,Y
4480 IF ANG<180 THEN 4600
4490 LOCATE X,Y+1,C
4500 IF C<>0 THEN 4600
4510 AS=0
4520 FOR W=1 TO 45
4530 LOCATE X,Y+W,QW
4540 IF QW<>0 THEN AS=1
4550 IF AS=1 THEN 4590
4560 COLOR Q
4570 PLOT X,Y+W
4580 Q=Q+1:IF Q>8 THEN Q=2
4590 NEXT W
4600 NEXT ANG
4610 A=80:P=0
4620 TRAP 9000
4630 FOR ANG=0 TO 359 STEP 10
4640 Z=PEEK(712)
4650 FOR I=712 TO 707 STEP -1
4660 POKE I,PEEK(I-1)
4670 NEXT I
4680 POKE 706,Z
4690 X=INT(((A*SINE(ANG+90))/3)+60)
4700 Y=INT((A*SINE(ANG))+125)
4710 U=USR(ADR(MP$),1,P,X,Y)
4720 A=A-0.5:P=P+1:IF P>8 THEN P=0
4730 IF X=48 AND Y=125 THEN 4760
4740 NEXT ANG
4750 GOTO 4630
4760 FOR I=X TO 128
4762 IF I>68 AND I<84 THEN 4800
4770 U=USR(ADR(MP$),1,P,I,Y)
4780 GOTO 4840
4800 U=USR(ADR(MP$),1,P,0,0)
4840 U=USR(ADR(WT$),4)
4850 NEXT I
4852 U=USR(ADR(MP$),1,P,0,0)
4860 U=USR(ADR(WT$),300)
9000 POKE 106,128:GRAPHICS 0
9020 GRAPHICS 2+16
9030 POSITION 4,5
9040 ? #6;"GOOD GRIEF!"
9050 U=USR(ADR(WT$),500)
9060 GRAPHICS 2+16:POSITION 0,4
9070 ? #6;"  RETURN TO REPEAT"
9080 ? #6;"
9090 ? #6;"    PUSH SPACE BAR "
9100 ? #6;"    TO RUN  ELEVEN "
9110 POKE 764,255
9120 K=PEEK(764):IF K=255 THEN 9120
9130 POKE 764,255:IF K=12 THEN RUN
9140 RUN "D:ELEVEN"
```

ELEVEN

```
1000 REM ELEVEN
1010 GRAPHICS 11
1020 FOR I=4 TO 68 STEP 4
1030 COLOR I/4
1040 PLOT I,0:DRAWTO I,180
1042 PLOT I+1,0:DRAWTO I+1,180
1044 PLOT I+2,0:DRAWTO I+2,180
1046 PLOT I+3,0:DRAWTO I+3,180
1050 NEXT I
1060 FOR W=1 TO 1000:NEXT W
1070 GRAPHICS 2+16:POSITION 0,4
1080 ? #6;"  RETURN TO REPEAT"
1090 ? #6;"
1100 ? #6;"  PUSH SPACE BAR  "
1110 ? #6;"    TO LOAD WINDOW "
1120 POKE 764,255
1130 K=PEEK(764):IF K=255 THEN 1130
1140 POKE 764,255:IF K=12 THEN RUN
1150 RUN "D:WINDOW"
```

WINDOW

```
1000 REM WINDOW
2000 REM __Assembly routines
2010 REM
2020 DIM WT$(57):REM __TDELAY.OBJ
2080 DIM M$(80)
2090 TRAP 2330
2100 GRAPHICS 8:REM _created window
2110 POKE 87,11:REM _fool OS
2120 POKE 623,192:POKE 752,1
2130 SETCOLOR 4,0,4
2140 SETCOLOR 2,0,0:SETCOLOR 1,0,10
2150 DL=PEEK(560)+PEEK(561)*256
2160 POKE DL+166,15+128
2170 HI=INT(ADR(D$)/256):POKE 513,HI
2180 LO=ADR(D$)-(HI*256):POKE 512,LO
2190 POKE 54286,192:POKE 764,255
2200 REM __Draw color bars
2210 FOR I=10 TO 160 STEP 10
2220 COLOR I/10
2230 FOR J=1 TO 9
2240 PLOT 0,J+I:DRAWTO 79,J+I
2250 NEXT J
2260 NEXT I
2270 REM __Messages
2280 GOSUB 8010
2290 GOSUB 8010
2336 POKE 764,255
2337 K=PEEK(764):IF K=255 THEN 2337
2338 POKE 764,255:IF K=12 THEN RUN
2339 POKE 54286,64:GRAPHICS 0
2340 GRAPHICS 2+16:POSITION 0,6
2341 ? #6;"    LOADING GEDIT"
2342 RUN "D:GEDIT"
2343 DATA ALL THAT COLOR AND A
2350 DATA TEXT WINDOW TOO!,)
2360 DATA RETURN TO REPEAT
2370 DATA OR SPACE BAR TO LOAD GEDIT,
8000 REM __Voice of PROTO
8010 ? "}":POKE 752,1:TD=0
8020 READ M$
8022 IF M$="}" OR M$="" THEN 8100
8030 L=LEN(M$):SP=(40-L)/2:TD=TD+SP
8040 POKE 657,SP
8050 FOR V=1 TO L
8060 SOUND 0,ASC(M$(V,V)),10,4
8070 ? M$(V,V);
8080 NEXT V:? :SOUND 0,0,0,0
8090 GOTO 8020
8100 X=USR(ADR(WT$),TD*20)
8110 RETURN
```



GEDIT

```
1000 REM GEDIT
2000 REM __Assembly routines
2010 REM
2020 DIM WT$(57):REM __TDELAY.OBJ
2060 REM
2070 DIM FM$(54):REM __FILMEM.OBJ
2110 REM
2120 DIM MM$(49):REM __MOVMEM.OBJ
2160 REM
2170 DIM SS$(95):REM __SOUNDS.OBJ
2230 REM
2240 DIM CC$(37):REM __CIOCTL.OBJ
2270 REM
4000 REM
4010 REM MAINLINE
4020 DIM F$(16):F$="D:TEMP.DRW"
4030 DIM H(10)
4040 GRAPHICS 10
4050 SA=PEEK(88)+PEEK(89)*256
4060 RESTORE
4070 REM SET UP COLORS
4080 FOR I=704 TO 712
4090 READ B:POKE I,B
4100 NEXT I
4110 DATA 12,0,29,66,71
4120 DATA 123,156,214,234
4130 REM DRAW SCREEN
4140 COLOR 1
4150 PLOT 0,0:DRAWTO 79,0
4160 PLOT 0,182:DRAWTO 79,182
4170 FOR I=1 TO 8
4180 COLOR I:C=(I*10)-1
```

```
4190 FOR J=184 TO 188
4200 PLOT C-1,J:DRAWTO C-9,J
4210 NEXT J
4220 NEXT I
4230 C=37:R=90:UC=1:N=1:CI=2:RI=3
4240 REM GET USER INPUT
4250 X=USR(ADR(WT$),40)
4260 TRAP 5170:POKE 764,255
4270 T=STRIG(0):S=STICK(0)
4280 FOR I=1 TO 9
4290 H(I)=PEEK(703+I)
4300 NEXT I
4310 K=PEEK(764):IF T=1 THEN 4370
4320 REM PAINT
4330 FOR I=R-RI TO R
4340 PLOT C,I:DRAWTO C+CI,I
4350 NEXT I
4360 REM FLASH CURSOR
4370 LOCATE C,R,C1
4380 LOCATE C,R-RI,C2
4390 LOCATE C+CI,R-RI,C3
4400 LOCATE C+CI,R,C4
4410 COLOR 0:PLOT C,R
4420 PLOT C,R-RI
4430 PLOT C+CI,R-RI
4440 PLOT C+CI,R
4450 X=USR(ADR(WT$),1)
4460 COLOR 1
4470 X=USR(ADR(WT$),1)
4480 PLOT C,R
4490 PLOT C,R-RI
4500 PLOT C+CI,R-RI
4510 PLOT C+CI,R
4520 X=USR(ADR(WT$),1)
4530 COLOR C1:PLOT C,R
4540 COLOR C2:PLOT C,R-RI
4550 COLOR C3:PLOT C+CI,R-RI
4560 COLOR C4:PLOT C+CI,R
4570 COLOR N
4580 REM CHECK KEYBOARD
4590 IF K=255 THEN 5190
4600 IF K=47 THEN POKE 764,255:END
4610 IF K=42 THEN N=0:GOTO 4250
4620 IF K=118 THEN 4040
4630 IF K<128 THEN 4750
4640 IF K=142 THEN RI=RI+1
4650 IF K=143 THEN RI=RI-1
4660 IF K=134 THEN CI=CI-1
4670 IF K=135 THEN CI=CI+1
4680 S=1:IF RI<0 THEN RI=0:S=2
4690 IF CI<0 THEN CI=0:S=2
4700 IF RI>20 THEN RI=20:S=2
4710 IF CI>20 THEN CI=20:S=2
4720 X=USR(ADR(SS$),S)
4730 GOTO 5290
4740 REM GET COLOR
4750 IF K<>18 THEN 5020
4760 N=INT((UC+10)/10):COLOR N
4770 COLOR 1
4780 PLOT UC-1,191:DRAWTO UC+7,191
4790 X=USR(ADR(WT$),40):COLOR W
4800 T=STRIG(0):IF T=0 THEN 4920
4810 S=STICK(0)
4820 REM __Color hue
4830 IF S<>13 AND S<>14 THEN 4910
4840 A=(N-1)*16
4850 IF S=14 THEN B=PEEK(704+N)+2
4860 IF S=13 THEN B=PEEK(704+N)-2
4870 IF B<0 THEN B=0
4880 IF B>255 THEN B=255
4890 POKE 704+N,B:GOTO 4790
4900 REM __Color selection
4910 IF S<>7 AND S<>11 THEN 4790
4920 COLOR 0
4930 PLOT UC-1,191:DRAWTO UC+7,191
4940 COLOR N
4950 IF T=0 THEN 4250
4960 IF S=7 THEN UC=UC+10
4970 IF S=11 THEN UC=UC-10
4980 IF UC<1 THEN UC=1
4990 IF UC>71 THEN UC=71
5000 GOTO 4760
5010 REM SAVE SCREEN
5020 IF K<>62 THEN 5100
5030 X=USR(ADR(SS$),2):POKE 764,255
5040 K=PEEK(764):IF K=255 THEN 5040
5050 IF K=47 THEN 4250
5060 CLOSE #1:OPEN #1,8,0,F$
5070 X=USR(ADR(CC$),1,11,SA,7680)
5080 CLOSE #1:GOTO 4250
5090 REM LOAD SCREEN
5100 IF K<>0 THEN 5190
5110 X=USR(ADR(SS$),2):POKE 764,255
5120 K=PEEK(764):IF K=255 THEN 5120
5130 IF K=47 THEN 4250
5140 CLOSE #1:OPEN #1,4,0,F$
5150 X=USR(ADR(CC$),1,7,SA,7680)
5160 CLOSE #1:GOTO 4250
5170 X=USR(ADR(SS$),3):GOTO 4250
5180 REM CURSOR MOVEMENT
5190 IF S=15 THEN 4270
5200 IF S=14 THEN R=R-RI-1
5210 IF S=13 THEN R=R+RI+1
5220 IF S=11 THEN C=C-CI-1
5230 IF S=10 THEN R=R-RI-1:C=C-CI-1
5240 IF S=9 THEN R=R+RI+1:C=C-CI-1
5250 IF S=7 THEN C=C+CI+1
5260 IF S=6 THEN R=R-RI-1:C=C+CI+1
5270 IF S=5 THEN R=R+RI+1:C=C+CI+1
5280 REM CHECK CURSOR LIMITS
5290 IF C<0 THEN C=0
5300 IF C>79-CI THEN C=79-CI
5310 IF R<1+RI THEN R=1+RI
5320 IF R>181 THEN R=181
5330 GOTO 4260
5340 END
```

GTIA GRAPHICS

by
Sheldon Leemon
with
Stan Gilbert

Take your COLOR, PLOT and DRAWTO programming commands to new levels of brilliance with this tutorial on the GTIA graphics chip in your Atari®! Go beyond Graphics Modes 1 through 8 and into Modes 9, 10 and 11, accessing nine new colors or 16 shades.

Learn from the many examples of graphics programming - including three-dimensional shapes and digitized pictures - and their listings! Proto will accompany you through Professor Von Chip's tutorial to help you understand the programming language needed to create all the special color effects possible from your Atari. You will then be able to take these hints and put them into your own programs to create the same results! Also included is a helpful GTIA Editor which enables you to create and save your own pictures using nine different colors on the screen AT THE SAME TIME!

Appropriate for all age levels of programmers

REQUIRES:
BASIC Cartridge
Joystick
16K tape or 32K disk
GTIA graphics chip

(Your computer will contain the GTIA chip if it was manufactured after January, 1982. To be sure, check for the 'G' in your serial number, or ask your Atari dealer.)

©1983 Educational Software Inc.

Atari is a registered trademark of Atari, Inc. ESI and Tricky Tutorial are trademarks of Educational Software Inc.

IF U > I THEN COLUR
THIS: IF THIS